VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VÝUKOVÁ APLIKACE PRO TRIGONOMETRII S VYUŽITÍM POČÍTAČOVÉ GRAFIKY

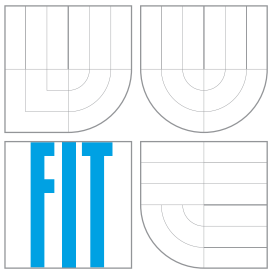
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN PLÍŠEK

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VÝUKOVÁ APLIKACE PRO TRIGONOMETRII S VYUŽITÍM POČÍTAČOVÉ GRAFIKY

EDUCATION COMPUTER PROGRAM FOR TRIGONOMETRY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN PLÍŠEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VÍT ŠTANCL

BRNO 2009

Abstrakt

Tato bakalářská práce se zabývá vyučováním matematických jevů, a to zejména výukou goniometrických funkcí pro druhý stupeň základních škol. Pro tyto účely vznikly 2 aplikace. První je aplikace *Goniometrie*, která se zabývá výukou goniometrických funkcí především pomocí zobrazení na pravoúhlém trojúhelníku, případně prezentací grafu funkce. Druhou aplikací je editor příkladů (*Editor*), kde lze jednoduše vytvořit příklad k dané funkci a následně ho při výuce použít přímo v aplikaci *Goniometrie*.

Abstract

This bachelor's thesis deals with teaching math's problems. Main objective is teaching goniometry functions for basic schools. There have been made two applications for these purposes. First application named *Goniometrie* is oriented to teach goniomety functions on projection onto rectangular triangle or by presentation graphs of these functions. Second application is example editor (*Editor*), which is designed to creating examples for application *Goniometrie*. These examples can be useful for teaching.

Klíčová slova

výuková aplikace, goniometrie, Pythagorova věta, graf funkce, tangens, cotangens, sinus, cosinus

Keywords

education software, goniometry, Pythagoras' theorem, graph of the function, tangens, cotangens, sinus, cosinus

Citace

Martin Plíšek: Výuková aplikace pro trigonometrii
s využitím počítačové grafiky, bakalářská práce, Brno, FIT VUT v Brně, 2009

Výuková aplikace pro trigonometrii s využitím počítačové grafiky

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Víta Štancla.

.....
Martin Plíšek
18. května 2009

Poděkování

Na tomto místě bych rád poděkoval svému vedoucímu, Ing. Vítu Štanclovi, za pečlivou kontrolu textů a věcné připomínky k řešenému projektu. Dále bych rád poděkoval paní učitelce Mgr. Jitce Kadlecové ze Základní Školy Krucemburk za konzultace ohledně matematické správnosti vyučovaných jevů.

Obsah

Kapitola 1

Úvod

Tento dokument popisuje návrh a implementaci aplikace sloužící k výukovým účelům, a to zejména k výuce matematiky na 2. stupni základních škol.

Cílem bylo navrhnout a implementovat aplikaci, která bude efektivním způsobem demonstrovat goniometrické funkce a zajistí, aby se žák při výuce co nejvíce soustředil na dané téma. Ke splnění těchto požadavků byly tedy vytvořeny dvě samostatné aplikace.

První aplikace – Goniometrie – je navržena jako aplikace výuková. Budou s ní tedy pracovat především žáci. Při návrhu bylo tedy vhodně zvoleno rozmístění prvků ve výukovém okně tak, aby měl žák všechny potřebné informace shromážděny u sebe. Aplikace demonstruje goniometrické funkce na pravoúhlém trojúhelníku a poté je možno zobrazit průběhy jednotlivých funkcí. Posledním prvkem této aplikace je možnost zobrazení příkladů. Tyto příklady jsou do aplikace dodávány jako externí soubory a jejich tvorbou se zabývá druhá aplikace.

Druhou aplikací je program Editor. Tato aplikace se zaměřuje na tvorbu příkladů, které mohou být využity k výuce přímo v druhé aplikaci Goniometrie. Editor je určen především pro kantory. Cílem tohoto programu je umožnit kantorovi efektivně a rychle vytvořit příklad pro aplikaci Goniometrie.

Tento dokument se skládá z několika částí. V úvodní kapitole 2 jsou popsány základní didaktické zásady, kterými jsme se řídili při návrhu a implementaci aplikací. Tato kapitola také obsahuje teorii matematických jevů, které jsou předmětem výuky. Další dvě kapitoly (3 a 4) se zabývají návrhem obou aplikací. Pro porovnání jsou zde zobrazeny náčrtky, podle kterých byly aplikace navrhovány a výsledné vzhledy oken v operačním systému Windows Vista. Následující dvě kapitoly (5 a 6) se zabývají vlastní implementací obou aplikací a zdůrazňují problémy, které během vývoje vznikaly a jak byly tyto problémy řešeny. Předposlední kapitola 7 se věnuje možným rozšířením, kterými by mohly být obě aplikace doplněny. Závěrečná kapitola shrnuje celkovou práci a poznatky.

Kapitola 2

Analýza problematiky

Naším úkolem bylo vytvořit aplikaci, která bude sloužit jako výukový software matematiky pro základní školy. Nejdříve jsme se tedy museli pozastavit nad tím, jak se matematické jevy na základní škole vyučují.

2.1 Úvod do didaktiky

Didaktiku jako takovou si zde nebudeme rozebírat, ale zaměříme se hlavně na tzv. didaktické zásady. Tyto zásady se také někdy označují jako zásady procesu vyučování – učení se. Dodržováním těchto zásad se dosáhne maximálního efektu při procesu učení. Všechny tyto zásady lze najít na [4] , my se zde ale budeme zabývat pouze zásadami, které jsou důležité pro vyučování pomocí výukového softwaru. Jsou to následující zásady:

Zásada uvědomělosti a aktivity

Podle této zásady by se měl učitel snažit své žáky zaujmout a udržovat s nimi kontakt a snažit se, aby si žáci uvědomili přínos probírané látky. Z našeho hlediska to znamená, že jsme se snažili naprogramovat aplikaci tak, aby žáka zaujmula. Pro lepší demonstraci probírané látky jsou situace zobrazovány na příkladech (Editor příkladů a jejich následná prezentace v programu Goniometrie).

Zásada komplexního rozvoje žáka

Základním kamenem této zásady je osobnost žáka. Tuto osobnost dělíme na 7 oborů: jazykový, matematicko-logický, vizuální/prostorový, hudební, interpersonální, intrapersonální a tělesný/fyzický. Pro naše účely je hlavně důležitý obor vizuální/prostorový (snažili jsme se navrhnout aplikaci tak, aby byly důležité části správně rozvrženy, barevně zvýrazněny, a aby bylo dobře navrženo uživatelské rozhraní) a matematicko-logický (vychází už z toho, že se jedná o výukový software matematiky).

Zásada spojení teorie s praxí

Tato zásada, jak již název napovídá, se zabývá tím, že učitel musí své žáky přesvědčit, že probírané učivo je pro ně přínosné a že ho v životě využijí. V naší aplikaci toto demonstrují příklady na jednotlivé funkce.

Zásada přiměřenosti

Zásada v důsledku říká, že je důležité odhadnout míru obtížnosti probírané látky. Například můžeme k látce prezentovat příklady vzestupné obtížnosti. Dále také říká, že látka na sebe musí navazovat. V tomto směru prezentujeme v programu jako prerekvizitu Pythagorovu větu.

Zásada individuálního přístupu

Zásada je založena na tempu chápání probírané látky žákem. Každý žák má individuální schopnost chápat probíranou problematiku. Proto v aplikaci používáme příklady jednodušší a v případě potřeby může učitel bystřejším žákům otevřít časově náročnější úlohy.

Zásada názornosti

Vychází z toho, že žáci již mají pravděpodobně o probírané látce nějaké představy (názor na věc), které je potřeba „vědecky“ vysvětlit. Názor může být zrakový, sluchový, čichový, chuťový, hmatový či pohybový. Logicky vyplývá, že my podporujeme zejména názor zrakový, a to vhodným zvýrazňováním důležitých věcí.

Zásada soustavnosti

Tato zásada vyžaduje, aby na sebe jednotlivé celky probírané látky logicky navazovaly a aby učební postup směřoval od jednodušší problematiky k problematice složitější. Toto, jak již bylo zmíněno, zajišťujeme zopakováním prerekvizitní znalosti, Pythagorovy věty.

2.2 Teorie potřebných matematických jevů

Teorie vyučovaná naším programem není příliš obsáhlá, ale spíš nám jde o to, abychom mohli tyto funkce popsat co nejjednodušeji, aby je žáci lehce zvládli. Proto se nejprve zabýváme zopakováním Pythagorovy věty a následně čtyřmi základními goniometrickými funkcemi. Teorémy popsané v této kapitole jsou ve stejném tvaru použity i v aplikaci (kde jsou navíc zvýrazněna klíčová slova).

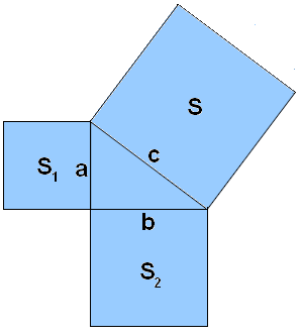
2.2.1 Pythagorova věta

Jelikož je Pythagorova věta, stejně jako goniometrické funkce, založena na pravoúhlém trojúhelníku, je velice žádoucí tuto látku žákům zopakovat. Pokud známe pro pravoúhlý trojúhelník dvě strany, respektive jejich délky, potom můžeme v trojúhelníku pomocí Pythagorovy věty dopočítat délku třetí strany.

Teorém:

> Obsah čtverce sestrojeného nad přeponou (nejdelší stranou) pravoúhlého trojúhelníka je roven součtu obsahů čtverců nad jeho odvěsnami (dvěma kratšími stranami).

Situace pro Pythagorovu větu je zobrazena na obrázku 2.1 a k tomuto obrázku se i vztahují vzorce (2.1, 2.2) pro výpočet třetí strany v pravoúhlém trojúhelníku.



Obrázek 2.1: Vyjádření Pythagorovy věty na pravoúhlém trojúhelníku

Vzorce Pythagorovy věty:

$$S = S_1 + S_2 \tag{2.1}$$
$$c^2 = a^2 + b^2 \tag{2.2}$$

2.2.2 Goniometrické funkce

Čtyři základní goniometrické funkce jsou tangens, cotangens, sinus a cosinus. Pro naše účely je důležité zobrazení funkcí na pravoúhlém trojúhelníku. Následně umožňujeme zobrazení grafů těchto funkcí.

Teorémy goniometrických funkcí

Tangens: Poměr délek protilehlé a přilehlé odvěsny v pravoúhlém trojúhelníku s ostrým úhlem α se nazývá tangens úhlu α – píšeme též $\tan\alpha$.

Cotangens: Poměr délek přilehlé a protilehlé odvěsny v pravoúhlém trojúhelníku s ostrým úhlem α se nazývá cotangens úhlu α – píšeme též $\cot\ \alpha$.

Sinus: Poměr délek protilehlé odvěsny a délky přepony v pravoúhlém trojúhelníku s ostrým úhlem α se nazývá sinus úhlu α – píšeme též $\sin\ \alpha$.

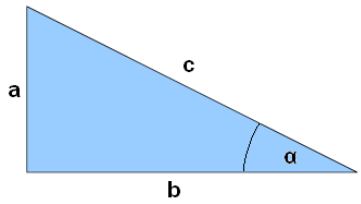
Cosinus: Poměr délek přilehlé odvěsny a délky přepony v pravoúhlém trojúhelníku s ostrým úhlem α se nazývá cosinus úhlu α – píšeme též $\cos\ \alpha$.

Vzorce pro goniometrické funkce

U každé takové funkce v aplikaci vypisujeme vzorec ve tvaru poměru dvou stran.

$$\tan\alpha = \frac{a}{b}$$
$$\cot\alpha = \frac{b}{a}$$
$$\sin\alpha = \frac{a}{c}$$
$$\cos\alpha = \frac{b}{c}$$

Pro lepší orientaci je na obrázku 2.2 zobrazen pravoúhlý trojúhelník s popsanými stranami.

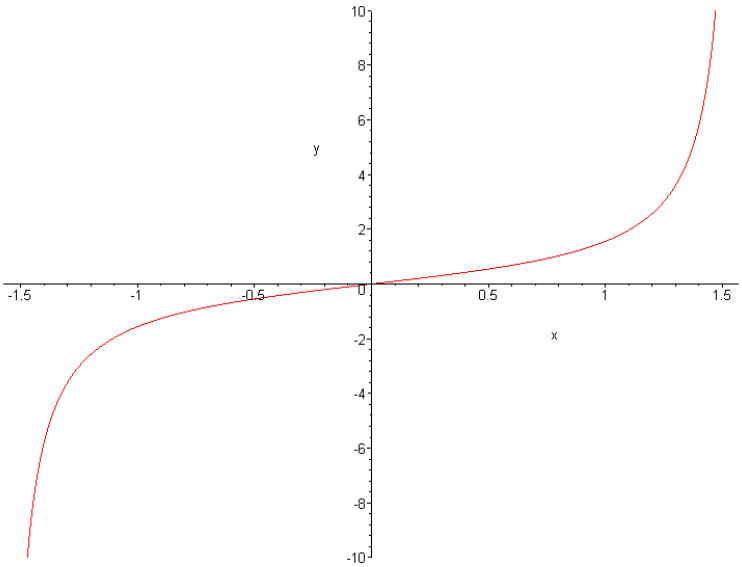


Obrázek 2.2: Pravoúhlý trojúhelník s ostrým úhlem α

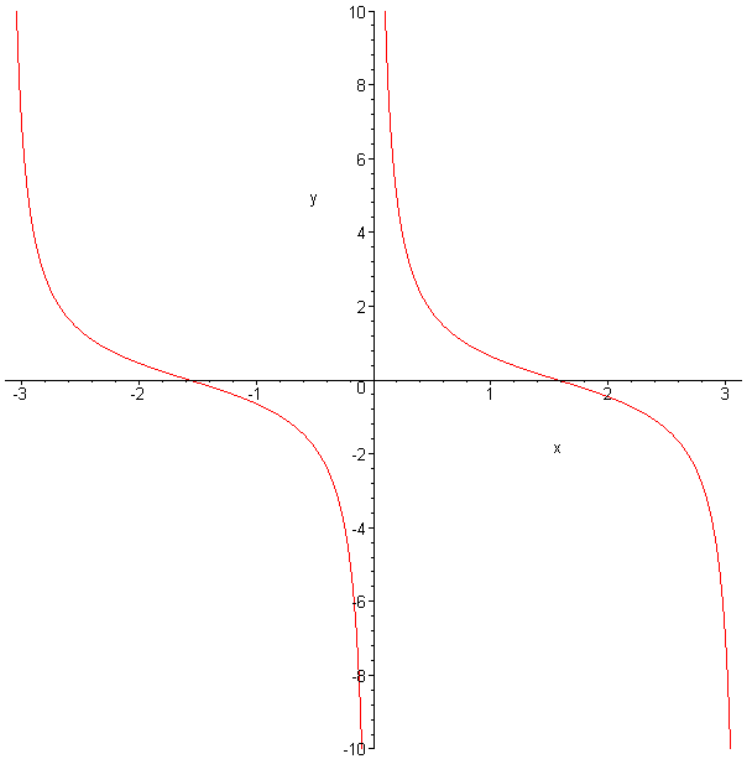
Grafy goniometrických funkcí

Poslední věcí, která nám zbývá pro ucelení znalostí o těchto goniometrických funkcích, jsou průběhy jednotlivých funkcí. Je obecně známo, že goniometrické funkce jsou periodické, což znamená, že se po určitém čase (periodě) průběh opakuje.

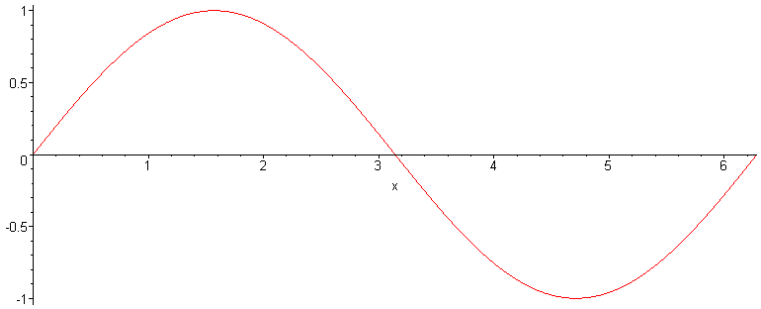
Na obrázku 2.3 je průběh funkce tangens, na obrázku 2.4 je průběh funkce cotangens, na obrázku 2.5 průběh funkce sinus a na obrázku 2.6 průběh funkce cosinus. U těchto funkcí je jedna zvláštnost. Funkce tangens a cotangens jsou si podobné a mají takovou vlastnost, že pro jisté hodnoty x nabývají hodnoty ∞. Tyto body označujeme jako asymptoty a jsou v grafu (v programu) kresleny svislou čarou.



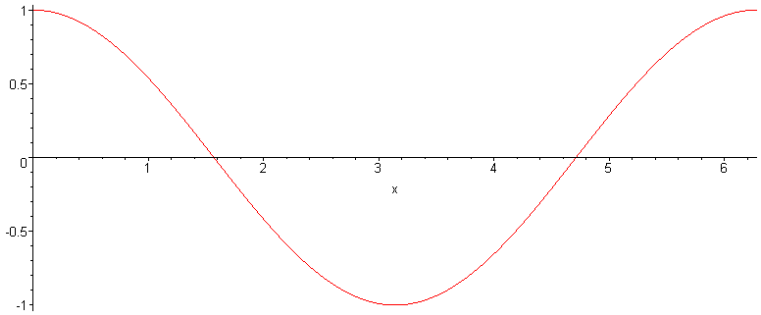
Obrázek 2.3: Průběh funkce tangens



Obrázek 2.4: Průběh funkce cotangens



Obrázek 2.5: Průběh fuknce sinus



Obrázek 2.6: Průběh funkce cosinus

Kapitola 3

Návrh řešení aplikace Goniometrie

Tato kapitola se věnuje návrhu aplikace Goniometrie, která bude sloužit jako výuková aplikace. První věc, nad kterou jsme se při návrhu pozastavili, bylo rozvržení grafického uživatelského rozhraní. Jelikož jsme aplikaci programovali v programovacím jazyce Java, který v sobě již obsahuje knihovny pro tvorbu uživatelského rozhraní, nepotřebovali jsme pro GUI žádné dodatečné knihovny. Nejdříve jsme si tedy navrhli GUI jednotlivých oken, ve kterých zobrazujeme informace. Následně jsme navrhli tok programu, tedy posloupnost akcí vedoucí výukovým programem a případná rozšíření vně hlavního výukového kurzu (zobrazení příkladů, samostatných kapitol, průběhů funkcí).

Poznamenejme ještě, že námi navržený program bude sloužit jako desktopová aplikace. Při spuštění aplikace program kontroluje rozlišení zobrazovacího zařízení. Pokud je toto rozlišení menší než 640×480 pixulů, potom aplikace upozorní na nízké rozlišení a aplikace se ukončí. Tato podmínka platí i pro aplikaci Editor.

3.1 Typy oken

Naše aplikace obsahuje několik typů oken. Okna jsme si vhodně rozdělili do několika skupin (z hlediska podobnosti), protože jsme toho mohli následně využít při implementaci. Například pro jednu třídu vznikají dvě různá okna díky rozlišení parametrem v konstruktoru třídy. Ale o tom až dále. Ještě bych rád zdůraznil, že u oken *Pythagorova věta* a *Fuknce* se buď mohou nebo nemusí objevit tlačítka `<Zpět` a `Další>`, a to podle toho, zda bude spuštěn celý kurz nebo pouze jedna kapitola (výběr z menu).

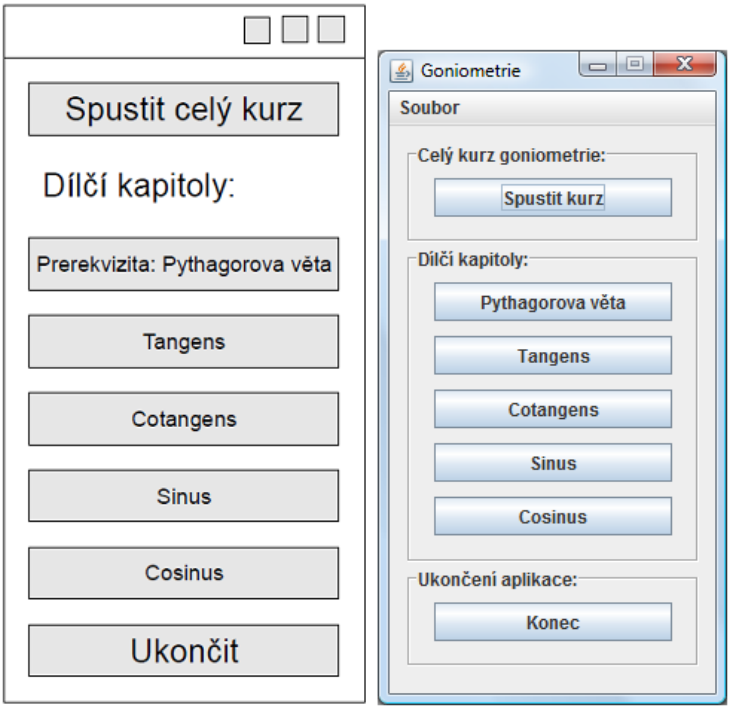
3.1.1 Hlavní menu

Snad každá aplikace v dnešní době zobrazí bezprostředně po svém spuštění hlavní menu. Naše aplikace není výjimkou. V hlavním menu je tlačítko pro spuštění celého kurzu goniometrických funkcí, případně jdou spustit samostatně vybrané kapitoly. Je zde samozřejmě i tlačítko pro ukončení aplikace. Podle tohoto popisu by tedy hlavní menu mohlo vypadat jako na obrázku 3.1. Pro srovnání je vedle uveden i skutečný vzhled tohoto okna v hotové aplikaci.

Posledním prvkem tohoto menu je horní lišta, která obsahuje rozbalovací menu *Soubor*, obsahující dvě položky, a to *O aplikaci* a *Konec*.

První položka – *O aplikaci* – bude obsahovat základní údaje o aplikaci a o autorovi.

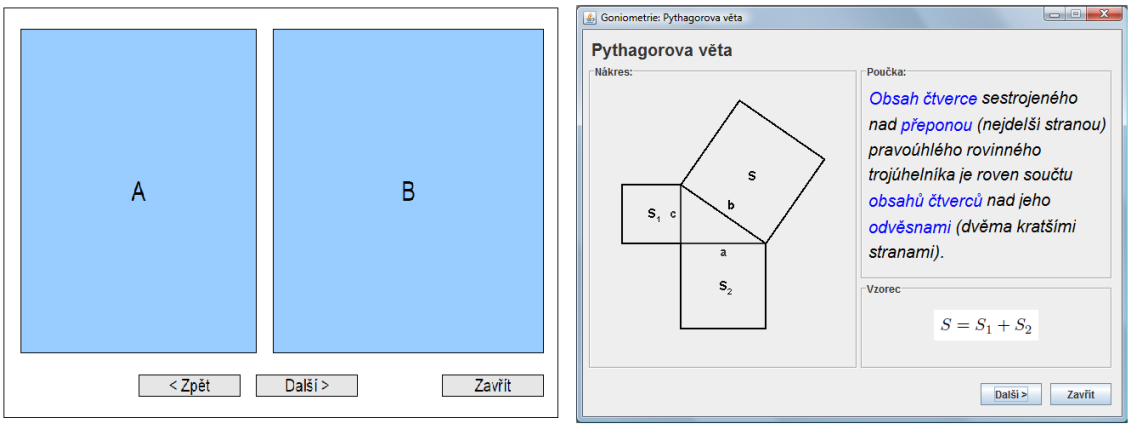
Poslední položkou menu *Soubor* je položka *Konec*. Vybráním této položky bude aplikace Goniometrie ukončena.



Obrázek 3.1: Návrh hlavního menu aplikace Goniometrie a skutečné menu v aplikaci

3.1.2 Pythagorova věta

V tomto okně se zobrazuje základní informace o Pytharovově teorému, který by žáci měli již znát. Slouží pouze pro zopakování. Rozvržení okna je definováno podle obrázku 3.2, který zobrazuje jak návrh tohoto okna, tak i jeho konečný vzhled v aplikaci.



Obrázek 3.2: Návrh okna pro zobrazení informací o Pythagorově větě a jeho skutečný vzhled

Oblast A slouží pro zobrazení teorému na pravoúhlém trojúhelníku a oblast B pro výpis poučky a vzorce. Náčrtek Pythagorovy věty na pravoúhlém trojúhelníku v oblasti A je interaktivní. Tento náčrtek koresponduje s poučkou o této větě. Pokud žák najede myší na slovo, které má v zobrazení Pythagorovy věty nějakou váhu, jako třeba slovo „přepona“,

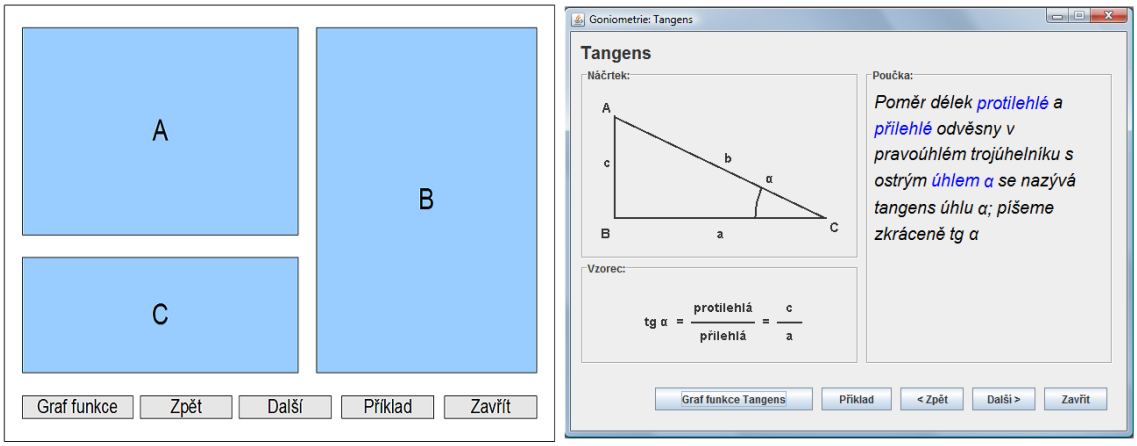
potom se v náčrtku přepona zvýrazní. Tento jev byl navržen na základě didaktické zásady komplexního rozvoje žáka.

Pokud analyzujeme Pythagorovu větu, potom bude vhodné zvýrazňovat při přejetí myší nad slovy, které jsou v následující citaci tučně:

> **Obsah čtverce** sestrojeného nad **přeponou** (nejdelší stranou) pravoúhlého trojúhelníka je roven součtu **obsahů čtverců** nad jeho **odvěsnami** (dvěma kratšími stranami).

3.1.3 Funkce

Toto okno (3.3) slouží pro zobrazování základních údajů o goniometrické funkci. Okno je rozděleno na několik logických podcelků. První oblast, označená A, slouží k zobrazení pravoúhlého trojúhelníku a popisků u něj. Druhá oblast – B – zahrnuje vzorec pro výpočet dané goniometrické funkce, v oblasti C budeme zobrazovat teorém funkce.



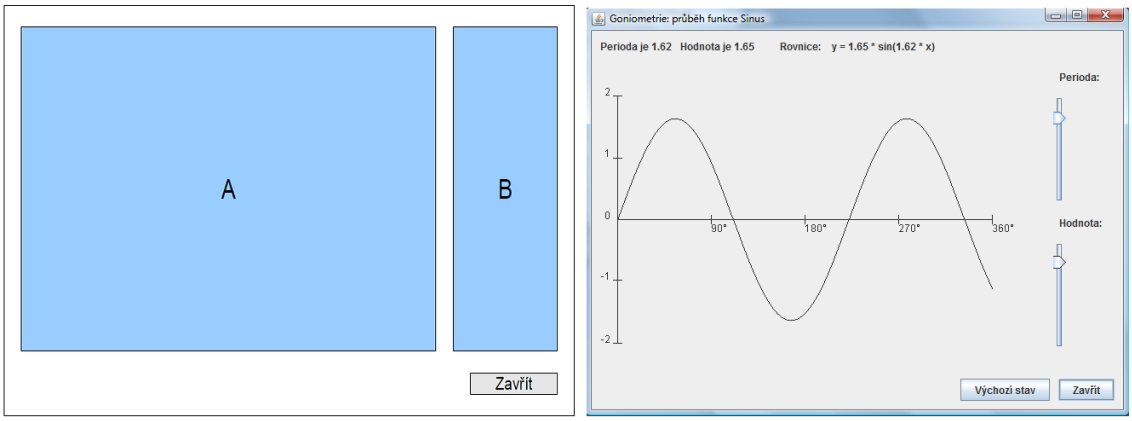
Obrázek 3.3: Návrh okna pro zobrazení informací o funkci a jeho skutečný vzhled

V tomto okně jsou také dvě důležitá tlačítka. Prvním je tlačítko `Graf funkce`, které po stisknutí zobrazí nové okno s průběhem funkce (viz. 3.1.3). Druhé tlačítko – `Příklad` – zobrazí okno, kde bude nějaký typový příklad k dané funkci. Více o tomto okně v kapitole 3.5. Důležité je ještě zdůraznit, že obě takto vyvolaná okna jsou chápána jako samostatná (stand-alone), tudíž samotný kurz/kapitola se po jejich spuštění nezavře, ale bude spuštěn pod novým oknem.

Průběh funkce

V tomto okně zobrazujeme průběh jednotlivých funkcí. Hlavním účelem tohoto okna je zobrazení variability grafů při různých parametrech – x, posunutí atd. Typově to vypadá tak, že po spuštění tohoto okna se vykreslí graf o základních parametrech. Poté bude možno tyto parametry funkce měnit pomocí tzv. trackbarů. Rozložení tohoto okna můžete vidět na obrázku 3.4. Jelikož změnu grafů žák řídí sám, proto je tento způsob zobrazení velice efektivní při procesu učení a vychází z didaktické zásady uvědomnělosti a aktivity.

Oblast A slouží pro vykreslení funkce, oblast B pro trackbary a vzorec s měnícími se koeficienty. V náčrtku tohoto okna ještě není zobrazena rovnice, která bude zobrazovat proměnné parametry u funkcí. Jak můžete vidět ve skutečném vzhledu tohoto okna, rovnice

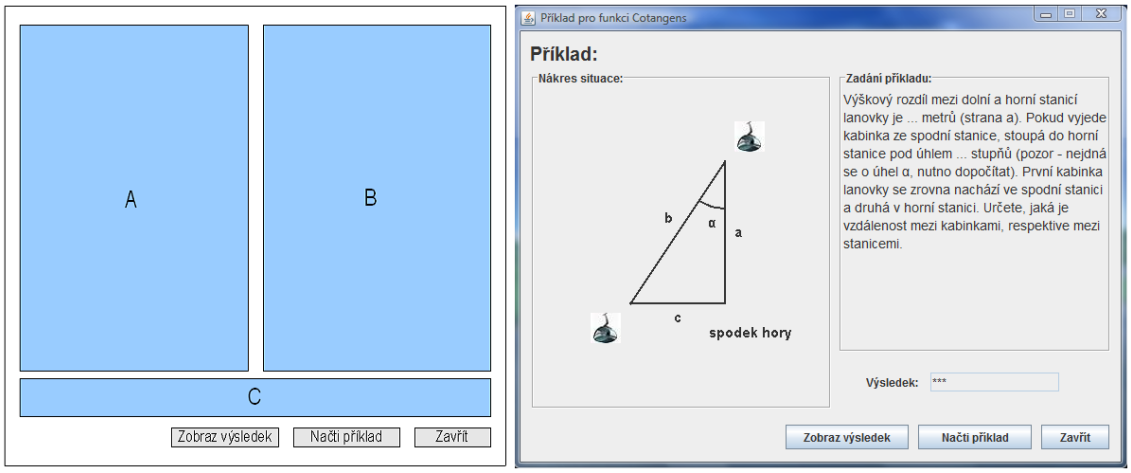


Obrázek 3.4: Návrh okna pro zobrazení průběhu funkce a jeho skutečný vzhled

byla nakonec umístěna v horní části okna. Ve skutečném vzhledu ještě přibylo tlačítko *Výchozí stav* pro nastavení posuvníků do výchozí polohy.

Příklad

Návrh okna pro zobrazení příkladu u funkcí a jeho konečný vzhled je na obrázku 3.5.



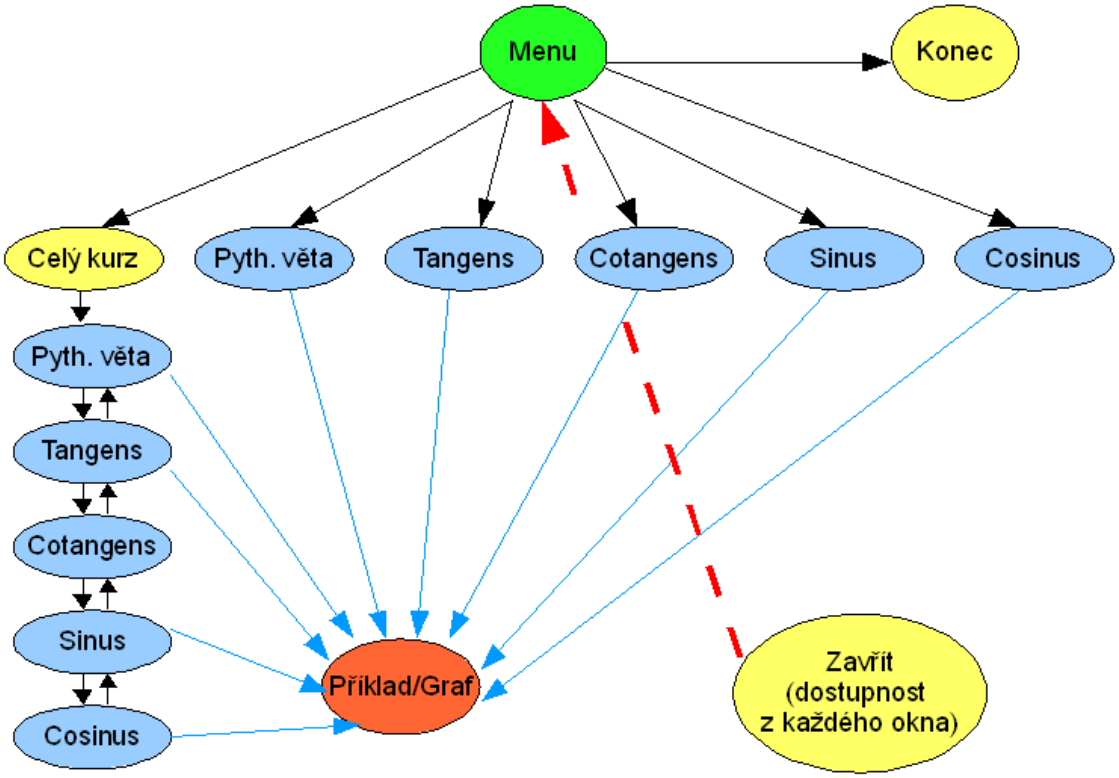
Obrázek 3.5: Návrh okna pro zobrazení příkladu a jeho vzhled ve skutečnosti

Oblast A obsahuje náčrtek pro příklad. V tomto náčrtku je zobrazen trojúhelník a k němu příslušné objekty na jednotlivých vrcholech, viz kapitola 4. V oblasti B bude umístěno zadání příkladu, které bude korespondovat s náčrtkem. V zadání budou popsány strany trojúhelníku a celková situace. V oblasti C je zobrazeno textové pole s výsledkem příkladu. Toto pole bude skryto, respektive bude předem vyplněno znaky `***`. K odhalení výsledku je dole v okně tlačítko `Zobraz výsledek`. Dalším tlačítkem v okně, `Načti příklad`, si bude moci žák otevřít příklad podle instrukcí učitele. Posledním tlačítkem – `Zavřít` – se okno s příkladem zavře. Důležité je si uvědomit, že obsah tohoto okna bude závislý na typovém příkladu přiřazenému k dané funkci. Abychom se drželi didaktických zásad, především zásady přiměřenosti a zásady individuálního přístupu, tak jako typový příklad zvolíme nějaký opravdu

jednuduchý příklad, aby ho zvládli i slabší žáci. Samotná přítomnost příkladů v aplikaci plní didaktickou zásadu spojení teorie s praxí.

3.2 Tok programu Goniometrie

Tato kapitola se zabývá posloupností akcí v programu, tak jak za sebou můžou následovat. Tyto akce jsou zobrazeny grafem na obrázku 3.6. Po spuštění aplikace se spustí hlavní menu. Z menu můžeme buď spustit celý kurz nebo jen jednotlivé kapitoly. Jednotlivá okna jsou v grafu reprezentována modrou barvou. Přechody mezi okny jsou reprezentovány černými šipkami. Z některých oken je možno zobrazit nezávislé okno (oranžovou barvou). Toto zobrazení je reprezentováno modrou šipkou a říká, že okno je vytvořeno „navíc“, tedy okno, ze kterého bylo toto okno spuštěno, nemizí, ale je zobrazeno pod oknem. Posledním prvkem v grafu jsou žluté bubliny, které zobrazují (pro větší přehlednost) pouze logické stavy. Speciální případ je okno menu, které je v aplikaci vždy a pokud vybereme některou možnost z menu, tak se nové okno zobrazí nemodálně, tudíž k menu bude stálý přístup. Pokud bude menu zavřeno, bude ukončena i celá aplikace.



Obrázek 3.6: Graf toku programu Goniometrie

Kapitola 4

Návrh řešení aplikace Editor příkladů

Tato kapitola popisuje návrh druhé aplikace, která slouží pro tvorbu příkladů k aplikaci Goniometrie. Kapitola obsahuje popis struktury této aplikace z hlediska vzhledu a funkčnosti a také popisuje formát souboru s příkladem.

4.1 Struktura aplikace

Naším cílem bylo vytvořit takovou aplikaci, aby v ní bylo možno jednoduše vytvářet příklady. Příklad by měl obsahovat následující prvky:

- náčrtek příkladu;
- popis příkladu;
- výsledek příkladu.

Prvním prvkem je náčrtek příkladu. Z hlediska didaktických zásad popsaných v kapitole 2.1 se jedná o nejdůležitější prvek příkladu, a proto jsme mu při návrhu věnovali největší váhu. Jelikož se jedná o příklady pro goniometrické funkce, které jsou definovány na pravoúhlém trojúhelníku, rozhodli jsme jako nejlepší základ pro náčrtek prvotní vykreslení tohoto trojúhelníku. Většinou jsou takové příklady specifické tím, jakými objekty jsou definovány jednotlivé vrcholy trojúhelníku. Například si představme typický příklad pozorovatele, který se dívá na vrchol nějaké budovy. Žák má například spočítat výšku této budovy a ze zadaní zná vzdálenost pozorovatele od budovy a úhel, pod kterým se pozorovatel dívá na vrchol budovy. Toto je základní příklad na funkce tangens a cotangens. Hlavními prvky v tomto příkladu jsou objekty pozorovatel, pata budovy a vrchol budovy. Samozřejmě tu zde hrají roli i strany v trojúhelníku, ale ty už nemají takovou váhu jako samotné vrcholy. Proto uživateli, který bude pomocí našeho programu tvořit příklad, dáme možnost vybrat si ze seznamu objektů, které je možné umístit na vrcholy trojúhelníku. Samozřejmě nemáme neomezené množství těchto objektů, a proto jsme vybrali pouze několik základních objektů. Pokud uživatel objekt v seznamu nenalezne, je v seznamu možnost *jednoduchý popisek*, díky níž je uživatel schopen popsat objekt slovy, samozřejmě pouze dvěma, třemi slovy. Strany trojúhelníku jsme tedy neuvažovali pro editaci a nepevno jsme je označili popisky a, b, a c. Pro případné popsání stran může uživatel použít pole zadání příkladu.

Dalším důležitým prvkem je popis příkladu, který bude zadáván do pole zadání příkladu. Zde uživatel napíše zadání příkladu a případně popíše i strany trojúhelníku viz. předchozí odstavec. Jelikož je většinou v příkladě zadán úhel nebo je zde alespoň popsán, aby byl spočítán, proto jsme nad polem zadání příkladu umístili tlačítko pro vkládání znaku α, jelikož tento znak nelze na klávesnici jednoduše zapsat.

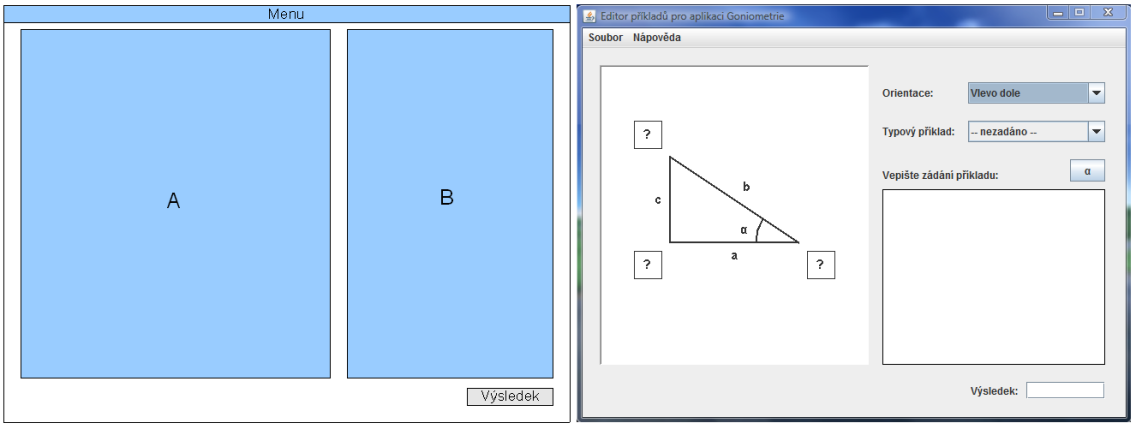
Posledním důležitým prvkem příkladu je výsledek. Výsledek bude zapsán v samostatném poli, čili nebude obsahem zádání. Toho je využíváno k tomu, že výsledek není zobrazen okamžitě, ale je možné ho zobrazit až po stisku tlačítka.

Dále jsme počítali s tzv. typovými příklady. Pokud chce třeba učitel vytvořit pro žáky příklad, ale nechce ho psát celý znovu, ale chce pouze zapsat hodnoty potřebné pro výpočet, potom by měla aplikace obsahovat některé typové příklady, kde je již předem definovaný náčrtek pro příklad, předepsáno zadání příkladu, pouze tedy bez hodnot potřebných k výpočtu, a učitel tyto hodnoty doplní.

Zatím jsme vždy uvažovali, že trojúhelník v náčrtku bude orientovaný pouze na jednu stranu. Pro příklady je však nutné, aby mohl být trojúhelník otočen. Proto jsme zde ještě vhodně přidali prvek pro výběr orientace trojúhelníku. Pro popsání většiny příkladů postačí, že lze trojúhelník otočit o 90°. Zrcadlové otočení neuvažujeme, protože je to zbytečné, jelikož objekty na vrcholech trojúhelníku mohou být měněny.

4.2 Hlavní okno

Tato sekce popisuje, jak byl navrhován vzhled aplikace Editor. Návrh vzhledu je zobrazen na obrázku 4.1, kde je poté také zobrazen konečný vzhled aplikace.



Obrázek 4.1: Vzhled aplikace Editor (návrh a konečná podoba)

Obsah okna

Náčrtek trojúhelníku pro příklad je zobrazen v oblasti A na obrázku 4.1. Tento náčrtek obsahuje samotný trojúhelník s popsanými stranami a, b, a c. Na vrcholech trojúhelníku jsou čtverečkem zobrazeny oblasti, do kterých se budou vykreslovat jednotlivé objekty. Pokud pro tyto oblasti není vybrán objekt, je v nich vykreslen otazník. Po kliknutí do oblasti bude uživateli zobrazen seznam objektů, které mohou být k vrcholu přiřazeny a později i vykresleny. Jak je popsáno výše, lze k vrcholu umístit i jednoduchý popisek.

Oblast B na obrázku 4.1 obsahuje následující prvky:

- roletové menu pro výběr orientace trojúhelníku;
- roletové menu pro případný výběr typového příkladu;
- textovou oblast pro vepsání zadání příkladu;
- textové pole pro vepsání výsledku příkladu.

Roletové menu pro výběr orientace trojúhelníku poskytuje 4 základní orientace. Vždy se bere orientace vzhledem k pozici pravého úhlu trojúhelníku. Základní trojúhelník má orientaci tzv. vlevo dole. Další možné pozice jsou vlevo nahoře, vpravo nahoře a vpravo dole.

Roletové menu pro výběr typového příkladu umožňuje učiteli rychlou tvorbu příkladů pomocí předem definovaných údajů. Menu obsahuje typické příklady jako třeba příklad s pozorovatelem a budovou nebo s majákem a lodí.

Další prvek oblasti B je textová oblast pro vepsání zadání příkladu. Nad touto textovou oblastí jsme ještě umístili dodatečné tlačítko se znakem α, které po svém stisknutí vloží do textové oblasti znak α. Toto je důležité pro zadávání úhlů v zadání příkladu, jelikož na klávesnici není žádná klávesa pro vložení tohoto znaku.

Posledním přímo viditelným prvkem v GUI je textové pole pro výsledek. Do tohoto pole bude vepsán uživatelem výsledek příkladu.

Menu Soubor a Nápověda

Hlavní okno aplikace Editor obsahuje menu *Soubor*, ve kterém jsou položky *Nový příklad*, *Otevřít*, *Uložit* a *Konec*, a menu *Nápověda*, kde jsou položky *Návod k použití* a *O aplikaci*.

První položka je *Nový příklad*. Pokud uživatel vybere tuto položku, všechny prvky okna budou nastaveny do počátečního stavu, tedy náčrtek se překreslí tak, že trojúhelník bude mít orientaci vlevo dole (pravý úhel) a vrcholy trojúhelníku nebudou mít přiřazený žádný objekt nebo popisek. Textové pole zadání příkladu bude vymazáno a připraveno pro vepsání nového zadání. Stejně tak bude vymazáno textové pole výsledek. Museli jsme ještě zvážit možnost, že uživatel mohl před označením položky *Nový příklad* vybírat v roletovém menu typový příklad. Proto musíme ještě toto menu nastavit do počátečního stavu, tedy bez označeného typového příkladu. Dále jsme mysleli na možnost, kdy uživatel třeba nechtěl vybrat položku *Nový příklad* a pouze kliknul na špatnou položku. Proto je po kliknutí na tuto položku uživatel dotazán, zda chce skutečně vynulovat všechny prvky.

Položky *Otevřít* a *Uložit* mají podobný význam. Položka *Otevřít* umožní uživateli vybrat na disku soubor s příkladem. Tento soubor bude následně otevřen a budou z něj načtena důležitá data pro samotný příklad. Tato data korespondují s nastavením jednotlivých prvků okna. O struktuře příkladu bude pojednávat následující část této kapitoly, tedy 4.3. Položka *Uložit*, jak již bylo řečeno, má podobný význam jako položka *Otevřít*, avšak místo toho, aby program načetl data ze souboru a nastavil jednotlivé prvky okna, bude tato data číst z vlastností prvku okna a bude je zapisovat do souboru na disk.

Poslední položkou menu *Soubor* je položka *Konec*. Vybráním této položky je aplikace Editor ukončena.

V druhém menu je první položka *Návod k použití*. I když je aplikace navržena pro poměrně intuitivní ovládání, tak každá aplikace by měla obsahovat stručný manuál k použití. Tento manuál v několika krocích popisuje, jak jednoduše vytvořit příklad pro aplikaci Goniometrie.

Poslední položka – *O aplikaci* – obsahuje základní údaje o aplikaci a o autorovi.

4.3 Příklady

V této části si popíšeme, jak jsme zvolili ukládání příkladů pro aplikaci Goniometrie. Pro ukládání dat do souboru musíme zvolit nějaký formát, který bude možné jednoduše vytvářet, číst a modifikovat.

Z počátku byly uvažovány dvě možnosti pro ukládání. Prvním způsobem bylo ukládání do souboru zápisem výrazů `proměnná=hodnota`. Tento způsob by byl celkem vhodný v případě, kdy proměnná může nabývat pouze jednoho typu hodnot a nemusí řešit dodatečné atributy. Například pokud budeme chtít uložit do souboru informaci o objektu na jednom z vrcholů trojúhelníku a tím objektem bude popisek. Do souboru bychom tedy zapsali `vrcholA=popisek`, ale už nikde nemáme místo pro zápis samotného popisku, tedy textu, který popisek obsahuje.

Pro toto ukládání se tedy nabízí mnohem lepší a modernější způsob, a to ukládání do XML dokumentu. Vezměme si třeba výše zmíněný příklad s popiskem. V XML ho můžeme jednuduše zapsat pomocí:

```
<vrcholA textPopisku="pták">popisek</vrcholA>
```

Naproti tomu, pokud nepotřebujeme dodatečnou informaci k objektu vrcholu, napíšeme to pouze jako:

```
<vrcholA>pozorovatel</vrcholA>
```

Další výhoda u XML dokumentů je fakt, že existuje spousta odladěných knihoven pro jednoduché operace s XML dokumenty. Pro jazyk Java je to například knihovna Dom4j [1]. Pro naší aplikaci jsme tedy zvolili ukládání do XML dokumentů s využitím právě této knihovny.

Ukládané informace

Do našeho dokumentu budeme potřebovat uložit následující informace:

- zadání;
- vrcholy A, B, C;
- výsledek příkladu;
- orientaci trojúhelníku.

U informací o vrcholech trojúhelníku budeme ještě uvažovat dodatečné informace, jak to bylo popsáno výše. Přesnou strukturu XML dokumentu popisujeme až v kapitole o implementaci.

Kapitola 5

Implementace aplikace Goniometrie

Tato kapitola popisuje, jak byla implementována aplikace Goniometrie. Celá implementace aplikace je rozložena do několika zdrojových souborů. Každý zdrojový soubor obsahuje většinou jednu třídu. Výjimku tvoří třídy, které mají nastaveny posluchače, takže mají v sobě obsaženou vnořenou třídu, případně anonymní třídu. Speciálním případem je třída `Balik.java`, kde jsou implementovány metody, které jsou využity ve více zdrojových souborech.

Tato kapitola bude popisovat implementaci tak, že začneme popisem tříd tak, jak jsou „potřeba“ v běžící aplikaci. Začneme tedy třídou `Main.java`, se kterou přímo souvisí třída `Menu.java`. Tato třída vyvolá první okno aplikace, z něhož bude možno spouštět okna pro funkce (zdrojový soubor `Funkce.java`) a pro Pythagorovu větu (zdrojový soubor `PythagorovaVeta.java`).

5.1 Spuštění aplikace a hlavní menu

Po bezprostředním spuštění aplikace nejdříve dochází ke kontrole rozlišení zobrazovacího zařízení. Ke kontrole jsou využity vlastní statické metody ze třídy `Balik.java` (kapitola 5.6.1). Poté je jednoduchou podmínkou ověřeno rozlišení.

Pokud je tedy rozlišení ověřeno, program vytvoří objekt aplikace pomocí volání konstruktoru třídy `Menu.java`. Od tohoto okamžiku je aplikace řízena událostmi. Menu funguje jako rozcestník pro uživatele pomocí tlačítek. Obsluha těchto tlačítek je popsána v samostatné kapitole 5.6.2. Pří kliknutí na jednotlivá tlačítka je vytvořen objekt dané třídy. Například pří kliknutí na tlačítko *Tangens* se vytvoří instance třídy Funkce pomocí volání konstruktoru takto:

```
Funkce tangens = new Funkce(Balik.TANGENS, false, null);
```

První parametr je pro rozlišení, o kterou funkci se jedná. Druhým parametrem je proměnná typu `boolean` pro řízení celého kurzu. Pokud je tato proměnná nastavena na logickou hodnotu `true`, potom je spuštěn celý kurz. Pokud je nastavena na `false`, potom byla kapitola spuštěna samostatně. Posledním parametrem je proměnná typu `Funkce`. Ta slouží pro průchod celým kurzem, čili pokud je logická proměnná pro celý kurz nastavena na `true`, potom bude tato proměnná obsahovat odkaz na sebe sama (na okno).

5.2 Okno s Pythagorovou větou

Implementace okna s Pythagorovou větou je umístěna ve třídě `PythagorovaVeta.java`. Okno obsahuje náčrtek pro Pythagorovu větu, vypsanou poučku Pythagorovy věty a související vzorec.

5.2.1 Náčrtek

Náčrtek pro Pythagovovu větu je implementován v souboru `NacrtekPythVeta.java`. Tato třída dědí třídu `Canvas` a používáme jí pro kreslení grafických primitiv a vypisování řetězců. V konstruktoru této třídy se provedou výpočty potřebných bodů pro vykreslení čar. Tyto body jsou reprezentovány objekty třídy `Point`. Tyto body jsou pouze abstraktní, tedy pouze pro uložení vypočítaných hodnot. Vlastní vykreslování se vykonává pomocí metod třídy `Graphics2D`. Vykreslování čar a řetězců na plátno je vykonáváno v metodě `public void paint(Graphics g)`. Jak můžeme vidět, metoda používá pro kreslení grafický kontext `Graphics g`. Tento grafický kontext však nabízí pouze omezené množství metod pro kreslení, proto používáme pro kreslení rozšířený grafický kontext `Graphics2D`. Jelikož metoda `paint()` musí vždy v parametru obsahovat třídu `Graphics g`, musíme rozšířený grafický kontext vytvořit až uvnitř metody pro kreslení, a to následujícím způsobem:

```
Graphics2D gr = (Graphics2D)g;
```

Uvniř metody `paint()` je obsažen řídící logický mechanismus pro vykreslování čar různou barvou. To záleží především na tom, zda bylo z hlavního okna požadováno zvýraznění například odvěsny nebo přepony. Tento mechanismus má proměnné typu `boolean` a jsou to proměnné `zvyrCtverecNadPreponou`, `zvyrOvesny`, `zvyrCtverceNadOdvesnami` a `zvyrPreponu`. Jak již název napovídá, názvy těchto proměnných korespondují se zvýrazněním jednotlivých čar.

Pro nastavení řídících proměnných jsou ve třídě implementovány tyto metody:

- `public void zvyrazniOdvesny(boolean stav)`,
- `public void zvyrazniPreponu(boolean stav)`,
- `public void zvyrazniCtverecNadPreponou(boolean stav)`,
- `public void zvyrazniCtverecNadOdvesnami(boolean stav)`.

Metody pomocí logické proměnné `stav` nastavují nebo ruší vnitřní výše zmiňované řídící proměnné. Tyto metody i automaticky aktualizují vykreslovací plochu, tudíž je efekt nastavení proměnných a následné zvýraznění okamžité.

5.2.2 Poučka

Poučka není implementována v samostatné třídě, ale kód, který se stará o vypsání poučky je součástí třídy `PythagorovaVeta.java`. K vypsání poučky se nabízelo použití některé textové komponenty, která by byla schopna jednoduše a hromadně nastavovat parametry vypsaného textu. Bohužel však toto nevyhovovalo požadavkům, kdy my potřebujeme, aby některá slova dovolovala odposlouchávání na události myši.

Díky tomuto požadavku byl nakonec použit panel `JPanel` s nastaveným správcem rozmístění typu `FlowLayout`. Tento správce rozmístění zajišťuje, že při vkládání komponent

na panel se budou komponenty zarovnávat jako psaný text, tedy zleva doprava a odshora dolů. Komponenty vkládané na panel jsou typu `Slovo` a jsou implementovány ve třídě `Slovo.java`. Tento typ komponenty dědí třídu `JLabel` a doplňuje pouze estetický vzhled daného slova, tedy velikost, styl a barvu textu.

Takto tedy vytváříme jednotlivá slova a vkládáme je na panel a ona se automaticky zarovnávají jako text. U některých klíčových slov (viz kapitola 3.1.2) jsou nastaveni posluchači. O naslouchání se stará `MouseListener` a naslouchá událostem `mouseEntered` a `mouseExited`. Při obsluze těchto událostí se využívá metoda pro zjištění zdroje události `getSource()`. Při zjištění události je tedy rozpoznán zdroj a podle něj je zavolána metoda pro zvýraznění v náčrtku, viz předchozí část této kapitoly.

5.2.3 Vzorec

Jelikož je v tomto okně statický vzorec, který se nemění, není tento vzorec generovaný pomocí metod jazyka Java, ale je vkládán na panel jako obrázek. Tento obrázek je vygenerován pomocí typografického systému LaTeX. Vzorce u goniometrických funkcí jsou řešeny trošku jinak, ale o tom až dále.

5.3 Goniometrické funkce

Tato kapitola popisuje implementaci oken s goniometrickými funkcemi. Implementace je obsažena ve třídě `Funkce.java` a tato třída využívá objektů vycházejících ze tříd `Trojuhelnik` a `VzorecFunkce`. Z této třídy lze spustit okna pro příklad a pro graf funkce. Implementace těchto dvou oken bude rozebrána v následujících kapitolách.

Jak již tedy bylo popsáno v návrhu, okno obsahuje náčrtek trojúhelníku, poté výpis poučky a vzorec pro funkci.

5.3.1 Náčrtek

Implementace náčrtku pro goniometrické funkce je obsažena ve třídě `Trojuhelnik` a je umístěna ve zdrojovém souboru `Trojuhelnik.java`. Implementace je řešena stejně jako u náčrtku Pythagorovy věty (5.2.1) pouze jsou zde jiné řídící proměnné. V tomto náčrtku potřebujeme zvýrazňovat odvěsny, a to buď přilehlou nebo protilehlou, přeponu a ostrý úhel.

Body jsou zde opět přepočítány a následně uloženy do objektů typu `Point`. Vykreslování je znovu zajišťováno pomocí metody `paint()`, která uvnitř mění svůj grafický kontext, jak tomu bylo v kapitole 5.2.1. V tomto náčrtku je navíc ještě použita metoda `drawArc()` pro vykreslení oblouku označující ostrý úhel v trojúhelníku.

Řídícími proměnnými jsou zde proměnné `zvyrazniPrilehlou`, `zvyrazniProtilehlou`, `zvyrazniPreponu` a `zvyrazniUhel`. Tyto proměnné nastavují metody:

- `public void zvyrazniProtilehlou(boolean stav)`
- `public void zvyrazniPrilehlou(boolean stav)`
- `public void zvyrazniPreponu(boolean stav)`
- `public void zvyrazniUhel(boolean stav)`

5.3.2 Poučka

Poučka je opět řešena jako panel s vloženými komponentami, zarovnávanými stejně jako slova v textu. Nyní jsou posluchači nastaveny na slova: protilehlé, přilehlé, úhel α a přepony. Tato slova jsou v poučce označena modrou barvou. Pokud uživatel přes slovo přejede myší, zavolá se vnitřní metoda třídy `Trojuhelnik` a bude zvýrazněna část náčrtku, viz předchozí část kapitoly.

5.3.3 Vzorec pro funkce

Jelikož programovací jazyk Java neobsahuje žádné standardní knihovny pro tvorbu vzorců (myšleno jako objekt s podobnou strukturou zápisu jako v typografickém systému LaTeX) a na internetu jsme nenalezli žádné vhodné knihovny, proto jsme si naprogramovali vlastní třídu pro zobrazování vzorců. Tato třída je navržena tak, že dovoluje vypisovat vzorec pouze pro goniometrické funkce. Celý tento mechanismus je implementován v souboru `VzorecFunkce.java` a souvisí s ním třída `ParametryVzorce.java`.

Celý proces vytvoření vzorce začíná zavoláním konstruktoru, který má tvar:

```
public VzorecFunkce(int funkce, int sirka, int vyska)
```

Prvním parametrem je identifikátor pro rozlišení, o kterou funkci se jedná a vychází z konstant, které jsou definovány v pomocné třídě `Balik.java` (kapitola 5.6.1). Druhý a třetí parametr vyjadřuje rozměry plátna, na které je vzorec vykreslen. Toto je zde proto, že pokud by chtěla být aplikace přizpůsobena pro zobrazení na projektoru, aby šla jednodušeji upravit. My jsme tyto hodnoty nastavili napevno, jelikož okna aplikace mají také pevné rozměry. Po zavolání konstruktoru je vytvořen pomocný objekt třídy `ParametryVzorce`. Ten je naplněn proměnnými podle toho, která funkce byla konstruktorem rozlišena. Tato třída obsahuje parametry jako čitatel, jmenovatel atd.

Třída `VzorecFunkce.java` obsahuje tři hlavní metody:

- `public void paint(Graphics g)`,
- `private int vratSirkuSlova(String slovo, Graphics gr)`,
- `private int spocitejSirku(Graphics gr, ParametryVzorce vzorec)`.

První metoda se stará o vykreslení vzorce.

Druhá metoda počítá šířku slova na základě předaných hodnot v parametrech metody. Proměnná `gr` typu `Graphics` obsahuje grafický kontext, z něhož získáme přiřazený styl písma typu `Font`. Z této proměnné pomocí metody `getFontMetrics()` získáme metriky tohoto fontu a dále pomocí volání metod `getHeight()` a `charWidth(znak)` získáme výšku a šířku znaku v pixelech. Díky takto získané šířce znaku procházíme v cyklu celé slovo a nakonec vrátíme z metody součet těchto šířek, což je šířka daného slova.

Třetí metoda počítá šířku celého vzorce. Využívá metody pro spočítání šířky slova a uvažuje i mezery mezi jednotlivými slovy a znaky = atd. Šířka vzorce se počítá proto, aby byl vzorec zarovnaný na střed plátna.

5.4 Graf goniometrických funkcí

Tato kapitola popisuje implementaci průběhů goniometrických funkcí. Celé okno, ve kterém je zobrazen graf funkce, je implementováno v souboru `OknoPrubehFunkce.java`. Toto

okno obsahuje samotný průběch funkce (soubor `PrubehFunkce.java`), rovnici k dané funkci s pohyblivými parametry a nakonec ovládací prvky pro nastavování proměnných pro zobrazovaný graf.

V okně jsou vytvořeny na globální úrovni popisky pro hodnotu a pro periodu typu `JLabel`. Dále je pro okno vytvořen objekt třídy `PrubehFunkce`, který je samostatně popsán v kapitole 5.4.1. Hlavními prvky tohoto okna jsou bezesporu ovládací prvky, tzv. posuvníky. Tyto posuvníky jsou vytvořeny jako objekt typu `JSlider`. Oba posuvníky mají rozsah hodnot 0 – 200, přičemž pro naše účely to bude znamenat rozsah 0 – 2.00. Posuvníky mají přiřazen posluchač při změně hodnoty, tzv. `ChangeListener`. Jelikož jsou tyto posuvníky pouze dva, jejich obsluha je implementována přímo s definicí prvku pomocí anonymní třídy. Pokud dojde ke změně na posuvníku, potom jsou nastaveny parametry pro objekt třídy `PrubehFunkce` pro překreslení grafu s novými parametry a také jsou nastaveny nové popisky pro hodnotu, případně periodu.

5.4.1 Průběh funkce

Samostatný průběh funkce je reprezentován třídou `PrubehFunkce`. Tato třída je zděděna od třídy `Canvas` a je implementována v souboru `PrubehFunkce.java`. Konstruktor třídy slouží pro inicializaci proměnných ve třídě a pro rozlišení dané vykreslované funkce. Dále třída obsahuje dvě hlavní metody:

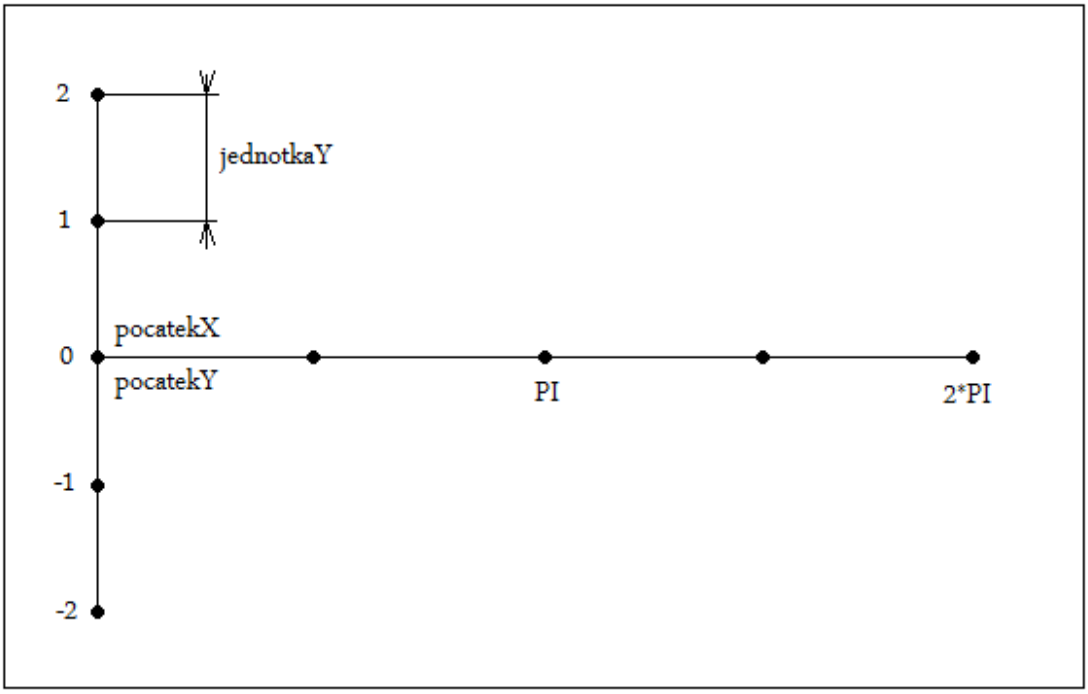
- `public void paint(Graphics gr)`,
- `public void nastavPerioduAHodnotu(double per, double hod)`.

Metoda paint je pomocí příkazu `switch` strukturována na čtyři části. Každá část obsahuje vlastní výpočty pro jednotlivou goniometrickou funkci a její následné vykreslení. Z hlediska vykreslování můžeme tyto čtyři funkce rozdělit do dvou skupin. První skupina obsahuje funkce tangens a cotangens, zatímco druhá skupina obsahuje funkce sinus a cosinus. První skupina je na vykreslování složitější, jelikož musíme hlídat, abychom nevykreslovali mimo vykreslovací plochu. Navíc funkce z této skupiny nabývají v některých hodnotách x obrovských hodnot, někdy i hodnot nekonečno. Oproti tomu druhá skupina obsahuje funkce, u kterých tento rozsah nemusíme hlídat, protože hodnoty jsou řízeny předanými hodnotami z posuvníků. Před vykreslováním funkcí jsou přepočítány počátky a nainicializovány počáteční hodnoty. Musíme si totiž uvědomit, že tyto funkce nabývají jak kladné, tak i záporné hodnoty a při vykreslování na plátno můžeme použít pouze kladné hodnoty. Proto počátek posuneme do prostředku vykreslovací plochy (myšleno vertikálně). Tento posunutý počátek je znázorněn obrázkem 5.1.

Skutečný počátek plátna, tak jak ho chápe java je v horním levém rohu. My si ho však přepočtem posuneme na souřadnice $[pocatekX, pocatekY]$. Proměnná $jednotkaY$ vyjadřuje hodnotu 1 na ose y. Vykreslovací algoritmus si popíšeme v následujících sekcích. Zde jen zmíníme, že je založený na posunování na ose x o hodnotu 1 pixel a spojování předchozí souřadnice a následující souřadnice čarou. Ještě je důležité upozornit, že musíme přepočítat hodnotu připadající na jeden pixel tak, aby spočítané hodnoty funkce skutečně korespondovaly s popisky na osách:

```
double pixel = 2 * Math.PI / delkaOsyX;
```

Druhá zmiňovaná důležitá metoda je `nastavPerioduAHodnotu(double per, double hod)`, která slouží pro nastavování periody a hodnoty u funkce, pokud na posuvníku dochází ke změně. Tím je zajištěna okamžitá aktualizace grafu.



Obrázek 5.1: Posunutí počátku pro průběh funkce

Tangens a cotangens

Nejprve musíme nainicializovat počáteční souřadnici pro vykreslování. U funkce tangens je tato souřadnice rovna počátečním souřadnicím, tedy *pocatekX* a *pocatekY*. U funkce cotangens je počáteční x-ová souřadnice rovna hodnote *pocatekX*, avšak y-ová souřadnice má v bodě 0 nedefinovanou hodnotu. Toto nám ale ošetří následující podmínka. V cyklu `for` tedy budeme počítat další souřadnice a spojovat je čarou s předchozími souřadnicemi. Ještě před spojením předchozí souřadnice s aktuální souřadnicí čarou je zkontrolováno, zda nová souřadnice y leží v rozsahu vykreslování, a to následovně:

```
if((y2 >= (pocatekY - 2 * jednotkaOsyY))
   && (y2 <= (pocatekY + 2 * jednotkaOsyY)))
```

Pokud je podmínka splněna, může být vykreslena spojnice těchto dvou souřadnic. Pokud však byla v minulém kroku souřadnice mimo rozsah, známe tedy jen jednu souřadnici a musíme tedy počkat jeden krok k vykreslení čáry (k nakreslení čáry potřebujeme 2 souřadnice). Na konci každého kroku je uložena aktuální souřadnice do proměnné pro předchozí souřadnici. Cyklus `for` má nastaven takový počet kroků jako je délka osy x v pixelech.

Nakonec ještě musíme spočítat, ve kterých hodnotách x budou vykresleny asymptoty. Jelikož víme, že funkce tangens a cotangens mají periodu rovnou hodnotě π a známe násobitel získaný z posuvníku nastavující periodu, potom můžeme jednoduše spočítat i hodnoty x pro asymptoty pomocí výrazu

```
asymptX = (int)((vykreslovanaPerioda / 2) + j * vykreslovanaPerioda);
```

kde *asymptX* je hodnota osy x, kde se nachází asymptota a j je iterační proměnná pro nalezení dalšího x. Algoritmus je samozřejmě ošetřen podmínkou pro průchod pouze po délce osy x.

5.4.2 Sinus a cosinus

Vykreslování funkcí sinus a cosinus je jednoduší než funkcí tangens a cotanges. Nemusíme zde totiž hlídat, zda jsou hodnoty y v rozsahu, jelikož obor hodnot funkcí je v intervalu $\langle -1; 1 \rangle$. Nemusíme se ani starat o vykreslování asymptot, jelikož funkce sinus a cosinus jsou definovány pro všechny hodnoty x.

Funkce sinus má počátek v bodě $[0, 0]$, tedy v bodě $[pocatekX, pocatekY]$. Funkce cosinus má počátek v bodě $[0, 1]$, takže musíme počátek pro y přepočítat pomocí výrazu:

```
y1 = (int)(-jednotkaOsyY * hodnota) + pocatekY;
```

Po nastavení inicializačních proměnných je pomocí cyklu `for` vykreslen průběh pro dané funkce.

5.5 Příklad

Implementace pro okno s příkladem je obsažena v souboru `Priklad.java`. V okně jsou objekty pro náčrtek situace příkladu (instance třídy `PrikladNakres`), zadání příkladu v textové oblasti typu `JTextArea`, textové pole typu `JTextField` pro zobrazení výsledku a poté také tlačítka pro načtení příkladu a zavření tohoto okna.

Při volání konstruktoru této třídy (`public Priklad(int funkce)`) bude pomocí proměnné funkce rozlišena goniometrická funkce a podle toho bude vybrán typový příklad. Tento příklad je uložený v XML dokumentu a je načítán pomocí třídy `XML.java`. Ta vychází, stejně jako třída pro náčrtek, z třídy používané v druhém programu, tedy z třídy v programu Editor. Jelikož je aplikace Editor založena na práci s XML dokumenty a s podobným náčrtkem pro příklad, bude tato problematika podrobněji popsána v kapitole 6. Tyto třídy jsou vlastně zjednodušené třídy převzaté z aplikace Editor, ale musejí tu být zastoupeny samostatně, jelikož se nacházejí v různých balíčcích a mají trošku jinou funkci.

5.6 Obecné programovací praktiky

Tato kapitola popisuje techniky, které jsou často v aplikaci používané, a které se ukázaly být správnou volbou.

5.6.1 Třída `Balik.java`

Tato třída obsahuje konstanty pro ostatní třídy, aby byla zachována jedinečnost proměnné a nevznikaly tzv. magické konstanty. Jsou zde obsaženy konstanty pro konstruktory těchto tříd:

- `Funkce.java`,
- `OknoPrubehFunkce.java`,
- `PrubehFunkce.java`,

- `VzorecFunkce.java`.

Tyto konstruktory většinou mají ve třídě větvící funkci, takže po zavolání konstruktoru s tímto parametrem bude vytvářený objekt přizpůsobený dané funkci (tangens, cotangens, sinus a cosinus). S těmito konstantami jsou ještě definovány konstanty pro speciální vnořené třídy obsluhující tlačítka (kapitola 5.6.2).

Dále třída obsahuje konstantu pro rozměr tlačítka v menu (Dimension) a dále konstanty pro styly fontů pro výpisy pouček v oknech popisujících funkce (Font).

Tuto třídu uzavírají tři statické metody:

- `public static String nazevFunkce(int funkce)`,
- `public static int vratSirkuObrazovky()`,
- `public static int vratVyskuObrazovky()`.

První metoda pomocí konstanty funkce předané přes parametr vrátí název funkce jako řetězec typu `String`. Toto se hodí při popisování oken jednotlivých funkcí. Druhá a třetí metoda vrací číslo typu `int` a vyjadřuje počet pixelů buď výšky nebo šířky obrazovky.

U této třídy je nutno podotknout, že třída nemá konstruktor a všechny prvky třídy jsou statické.

5.6.2 Obsluha tlačítek a položek menu

Skoro v každém okně, jak aplikace Goniometrie, tak i aplikace Editor, jsou umístěna různá tlačítka a menu, ale jejich obsluha je poněkud složitější. Během prací na těchto aplikacích bylo vyzkoušeno mnoho způsobů, jak tlačítka obsluhovat, ale následující způsob vypadal jako nepřehlednější a nejjednoduší.

Pokud okno obsahuje nějaká tlačítka nebo položky menu, potom třída obsahuje vnořenou třídu pro obsluhu tlačítek. Tyto třídy jsou pojmenovány různě, ale mají společné to, že implementují rozhraní `ActionListener`. Dále obsahují proměnnou, která se v konstruktoru inicializuje podle toho, jaká proměnná byla předána při nastavení posluchače. Základní struktura tedy vypadá takto:

```
class Obsluha implements ActionListener
{
    protected int idTlacitka;
    public Tlacitko(int id)
    {
        this.idTlacitka = id;
    }
    public void actionPerformed(ActionEvent e)
    {
        switch(idTlacitka)
        {
            case TL_1:
                // obsluha tlacitka 1
                break;
            case TL_2:
                // obsluha tlacitka 2
                break;
```

```
        }
    }
}
```

Toto je základní popis této třídy. Pokud chceme nyní nastavit obsluhu u tlačítka, které chceme označit pomocí proměnné TL_1, zavoláme u tlačítka následující metodu:

```
JButton tlacitko = new Tlacitko("Tlačítko TL_1");
tlacitko.addActionListener(new Obsluha(TL_1));
```

Nyní bude tlačítko nastaveno pro obsluhu událostí a při stisknutí tlačítka bude provedena obsluha ve vnořené třídě. Takto obsluhujeme události i při vybrání položky v menu.

5.6.3 Řízení celého kurzu

Dlouho jsme přemýšleli nad tím, jak jednoduše přepínat mezi okny aplikace, pokud byl spuštěn celý kurz.

Prvním nápadem bylo vytvořit na globální úrovni pole nebo spojový seznam a při přepnutí na další okno se posunout v seznamu. Tento způsob se však později nejevil jako správný a proto byl zvolen způsob, kdy je při přijití na další okno předán přes konstruktor objekt starého okna do okna nového. Tím se docílí toho, že všechna okna jsou spouštěna pouze jednou a po dobu běhu aplikace jsou aktivní v paměti. Mohlo by se zdát, že se tímto bude zbytečně zahlcovat pamět, ale profilování aplikace ve vývojovém prostředí Netbeans ukázalo, že aplikace zabírá při všech otevřených oknech něco kolem 60 MB operační paměti.

Kapitola 6

Implementace aplikace Editor

V této kapitole si popíšeme, jak byla implementována aplikace Editor a jakým způsobem byly ukládány příklady. Nejdřív popíšeme inicializační sekvenci při spuštění aplikace, poté stěžejní prvek celé aplikace, kterým je náčrtek příkladu a nakonec popíšemu způsob ukládání do XML dokumentů.

Všechny zdrojové kódy implementují rozhraní `Promenny`, které obsahuje konstanty potřebné pro řízení a identifikaci jednotlivých objektů.

6.1 Spuštění aplikace a inicializace

O spuštení aplikace se stará třída `Main.java`. V této třídě je definována kontrola rozlišení zobrazovacího zařízení stejně tak, jak je to popsáno v kapitole 5.1. Pokud rozlišení není dostatečné, uživatel bude upozorněn chybovou hláškou a aplikace bude ukončena. V opačném případě bude vytvořen objekt aplikace Editor zavoláním konstruktoru třídy `Editor.java` následovně:

```
Editor editor = new Editor();
```

Od této chvíle je aplikace inicializovaná a je řízena pomocí událostí.

6.2 Hlavní okno aplikace

Hlavní okno aplikace se skládá z několika viditelných objektů. Nejdůležitějším objektem je náčrtek příkladu, který je instancí třídy `Nakres.java`. Tato třída je popsána v samostatné kapitole, jelikož je poměrně rozsáhlá a obsahuje množství metod pro práci s ostatními prvky, především s XML objektem. Dalším prvkem je textová oblast, do které uživatel vepíše zadání příkladu. Tato oblast je definována výrazem:

```
private JTextArea zadaniPrikladu;
```

U této textové oblasti jsou ještě definovány parametry, aby byly zalamovány řádky a aby zalamovaná slova nebyla dělena, ale aby byla odřádkována na další řádek.

Dalšími dvěma prvky jsou roletové menu pro výběr orientace trojúhelníku v náčrtku a druhé roletové menu sloužící pro výběr typového příkladu. Obě menu jsou definována následovně:

```
private JComboBox vyberPrikladu;
String[] typyPrikladu = {"-- nezadáno --", "Lanovka",
        "Mrakodrap-pozorovatel", "Maják-loď", "Police"};
vyberPrikladu = new JComboBox(typyPrikladu);
private JComboBox vyberOrientace;
String[] typyOrientace = {"Vlevo dole", "Vlevo nahoře", "Vpravo nahoře",
        "Vpravo dole"};
vyberOrientace = new JComboBox(typyOrientace);
```

Posledním prvkem je textové pole pro zadání výsledku příkladu. Toto pole je nadefinováno pomocí:

```
private JTextField vysledek = new JTextField(8);
```

Hlavní okno ještě navíc obsahuje horní lištu s menu, která je popsána v kapitole 6.4.

6.3 Náčrtek příkladu

Vlastní náčrtek příkladu je implementován ve třídě `Nakres.java`, která dědí od třídy `Canvas`. Tato třída obsahuje několik typů vnitřních proměnných, která budou obsahovat důležitá data pro ukládání do XML souboru nebo data pro získání těchto dat.

Základ náčrtku tvoří body pro trojúhelník. Tyto body jsou definovány pomocí třídy `Point` a jsou rozděleny do dvou skupin. První skupinu tvoří body, které reprezentují vrcholy trojúhelníku a jsou to body `A`, `B` a `C`. Druhou skupinou jsou body, které popisují pozici popisku strany a jsou tedy umístěné přibližně mezi dvěma vrcholy. Tyto body jsou označeny `a`, `b` a `c`. Další proměnné, které reprezentují typ objektu na jednotlivých vrcholech trojúhelníku a jsou typu `int`, jsou `typObjektuA`, `typObjektuB` a `typObjektuC`. S těmito proměnnými korespondují oblasti, které jsou používány k naslouchání pro události myši. Tyto proměnné jsou typu `Rectangle` a ve zdrojovém kódu jsou označeny jako `Atype`, `Btype` a `Ctype`. Jsou definovány následovně:

```
private int paramOblasti = 32;
private Dimension oblast = new Dimension(paramOblasti, paramOblasti);
Rectangle Atype = new Rectangle(oblast);
Rectangle Btype = new Rectangle(oblast);
Rectangle Ctype = new Rectangle(oblast);
```

Tímto je prozatím nadefinován jejich rozměr, ale není ještě nadefinována jejich pozice. Pozice bude přepočítána na základě další důležité proměnné, a to `orientace`. Tato proměnná značí orientaci trojúhelníku podle pozice pravého úhlu. Poslední z hlavních proměnných jsou proměnné pro uložení popisků u vrcholů, které jsou typu `String` a ve zdrojovém kódu jsou označeny jako `popisekA`, `popisekB` a `popisekC`.

6.3.1 Volání konstruktoru

Samotný náčrtek se nainicializuje pomocí volání konstruktoru. Konstruktor má tvar:

```
public Nakres(int rozmer, int orientace, Editor okno)
```

kdy se nejprve nastaví vlastní vnitřní proměnné `rozmer`, `orientace` a `okno`. Proměnná `rozmer` reprezentuje velikost náčrtku. Šířka i výška náčrtku je stejná, a proto je zde takto předán pouze jeden rozměr. Proměnná `orientace` reprezentuje základní orientaci trojúhelníku. Posledním parametrem je proměnná `okno`, kterou zde potřebujeme kvůli pozdějšímu použití při volání modálních dialogů.

Dále je zde přidán posluchač pro naslouchání událostem myši. Tuto programovou konstrukci popisuje kapitola 6.3.3. Posledním inicializačním krokem je nastavení proměnných reprezentujících typ objektu na vrcholech na hodnotu `OBJ_ZADNY`.

6.3.2 Vykreslení trojúhelníku

Vlastní vykreslení trojúhelníku je implementováno v překryté metodě `paint()`. Tato metoda obsahuje jeden `switch` příkaz, který obsahuje čtyři sekce korespondující se čtyřmi druhy orientace trojúhelníku.

Podle druhu orientace (buď nainicializované pomocí konstruktoru nebo nastavené pomocí metody `zmenOrientaci(int orientace)`) se nastaví jednotlivé body potřebné pro vykreslování (`A`, `B`, `C`, `a`, `b` a `c`). Dále se nastaví souřadnice pro jednotlivé oblasti naslouchání událostem myši (`Atype`, `Btype` a `Ctype`), které jsou využity i při vykreslování objektu na vrcholy trojúhelníku. Dále jsou ještě v této sekci nastaveny vnitřní proměnné pro správné vykreslení oblouku u ostrého úhlu.

Nyní už jsou všechny hodnoty, díky orientaci, známé a proto můžeme vykreslit samotný trojúhelník pomocí metod grafického kontextu:

```
// nakreslime trojuhelnik
gr.drawLine(A.x, A.y, B.x, B.y);
gr.drawLine(B.x, B.y, C.x, C.y);
gr.drawLine(C.x, C.y, A.x, A.y);
```

Poslední akcí při vykreslování je vykreslení jednotlivých objektů na vrcholy trojúhelníku. Toto vykreslování objektů je vykonáváno metodou `kresliObjekt()` a je popsáno v samostatné kapitole 6.3.4.

6.3.3 Naslouchání událostem myši

Jelikož vykreslený náčrtek implementuje rozhraní `MouseListener`, musíme nějakým způsobem odchytávat události myši a následně tyto události osluhovat. Pro odchytávání událostí je v konstruktoru nastaven posluchač pomocí

```
this.addMouseListener(new Udalost());
```

Díky tomu je okamžitě vidět, že obsluha akcí bude prováděna pomocí třídy s hlavičkou:

```
public class Udalost implements MouseListener
```

Pokud tedy nastane událost myši, bude zavolána tato třída. Nás zajímá pouze jednoduché kliknutí a proto píšeme obsluhu do metody `mouseClicked(MouseEvent e)`. Ostatní události nás nezajímají, proto metody pro jejich obsluhu necháme prázdné. Pokud nastane událost, tak z proměnné `e` získáme souřadnice, kde tato událost nastala, a to pomocí metod `e.getX()` a `e.getY()`. Tyto souřadnice porovnáme se souřadnicemi jednotlivých oblastí pro naslouchání (`Atype`, `Btype` a `Ctype`). Pokud byla událost vyvolána v některé z těchto

oblastí, tak si specifický typ oblasti uložíme do proměnné a zavoláme dialog pro výběr objektu. Uživatel si zde vybere objekt, který chce umístit na daný vrchol a potvrdí. Tímto se zavolá vnitřní metoda třídy `Nakres` a nastaví se daný objekt takto:

```
nastavObjekt(oblast, prevedStringNaTypObjektu(volba), popisek);
```

Proměnná `oblast` specifikuje oblast, kam uživatel provedl kliknutí a pomocí vnitřní metody třídy obsluhující událost `prevedStringNaTypObjektu(volba)` je převeden řetězec reprezentující volbu na proměnnou typu `int` korespondující s rozhraním `Promenny`.

6.3.4 Vykreslování objektu na vrchol trojúhelníku

Vykreslování objektu je implementováno pomocí zavolání metody například takto:

```
kresliObjekt(gr, Atype, typObjektuA);
```

Takto je předán dovnitř metody grafický kontext, do kterého bude objekt vykreslen. Druhým parametrem je oblast u vrcholu, do které bude objekt vykreslen a poslední parametr je typ objektu. Metoda `kresliObjekt()` obsahuje velký příkaz `switch`, podle kterého je rozhodnuto, který objekt bude vykreslen.

Objekt je principiálně vykreslován pomocí následujících příkazů:

```
...
Rectangle misto = new Rectangle(oblast);
Icon obr;
...
case OBJ_LETADLO:
    obr = new ImageIcon(cesta + "letadlo.jpg");
    obr.paintIcon(this, gr, misto.x, misto.y);
    break;
...
```

6.3.5 Doprovodné metody

Dále tato třída obsahuje metody převážně pro získávání informací pro ukládání do XML dokumentů. Jsou to metody:

- `public int vratObjektA()`
- `public int vratObjektB()`
- `public int vratObjektC()`
- `public String vratPopisekA()`
- `public String vratPopisekB()`
- `public String vratPopisekC()`

Poslední doplňkovou metodou je metoda `spocitejDelkuPopisku()`, ketrá spočítá délku popisku v pixelech a umožňuje tak vykreslit popisek na střed.

6.4 Menu v hlavním okně

V menu Soubor jsou obsaženy položky pro vytvoření nového příkladu, uložení příkladu, otevření starého příkladu a ukončení aplikace. V menu Nápověda je položka pro zobrazení manuálu k aplikaci a položka pro zobrazení dialogu O aplikaci.

6.4.1 Menu Soubor

Pokud je z menu vybrána položka *Nový příklad*, aplikace se nejdříve dotáže, zda mají být skutečně nastaveny všechny prvky aplikace pro vytvoření nového příkladu. Pokud uživatel potvrdí tvorbu nového příkladu, bude aplikace uvedena do počátečního stavu. Pro objekt třídy `Nakres` bude zavolána metoda `inicializuj()`, jednotlivá roletová menu budou nastavena na základní indexovou položku a textová oblast pro zadání příkladu a textové pole pro zadání výsledku bude vymazáno pomocí volání metody `setText("")`. Pokud uživatel zruší akci, nic se nestane.

Pokud uživatel vybere položku *Otevřít...*, potom se mu zobrazí dialog pro výběr souboru pro otevření. Potvrzením volby bude objekt výběru nainicializován a můžeme z něj přečíst cestu k souboru:

```
JFileChooser otevrit = new JFileChooser();
...
otevrit.getSelectedFile().getPath();
```

Takto získáme cestu a jednoduše otevřeme XML soubor pomocí metod z třídy `XML.java`.

Stejným způsobem funguje dialog pro ukládání souborů, tedy pro výběr položky *Uložit...*. O detailech zápisu do souboru XML pojednává kapitola 6.5. O ukládání v obecném slova smyslu se stará metoda `ulozDoXML(String nazevSouboru)` ze třídy `Editor`.

Poslední položka – *Konec* – ukončuje aplikaci pomocí příkazu:

```
System.exit(0);
```

6.4.2 Menu Nápověda

Menu nápověda obsahuje dvě položky: *Návod k použití* a *O aplikaci*. Obě tyto položky po výběru zobrazí vlastní dialog. V prvním je popsán postup, pro jednoduchou tvorbu příkladu a v druhém informace o aplikaci.

6.5 Ukládání do XML dokumentu

Tato kapitola popisuje problematiku ukládání a načítání příkladů ve formátu XML do XML dokumentů. Pro ukládání a načítání jsou využity metody opensource knihovny Dom4j [1]. Implementace reprezentující operace s XML strukturou je obsažena ve zdrojovém souboru `XML.java`.

6.5.1 Struktura XML dokumentu

O struktuře XML dokumentu jsme se již nepatrně zmínili při návrhu apliakce Editor 4. Zde si popíšeme přesnou strukturu XML dokumentu. Zde je příklad uloženého XML dokumentu reprazentujícího příklad:

```
<?xml version="1.0" encoding="UTF-8"?>

<priklad>
  <zadani>test jestli to jde</zadani>
  <vysledek>ano</vysledek>
  <orientace>1</orientace>
  <bodA>1</bodA>
  <bodB>3</bodB>
  <bodC popisek="ahoj">6</bodC>
</priklad>
```

Příklad obsahuje všechny prvky, které může uložený XML dokument obsahovat. Element `<zadani>` příkladu obsahuje text získaný z textové oblasti pro zadání. Element `<vysledek>` obsahuje text získaný z textového pole pro výsledek příkladu. Element `<orientace>` obsahuje číslo typu `int` pro rozlišení orientace trojúhelníku. Poslední tři elementy (`<bodA>`, `<bodB>` a `<bodC>`) obsahují informaci o typu objektu na vrcholu trojúhelníku. Všimněme si, že element `<bodC>` obsahuje atribut `popisek="ahoj"`, protože typ objektu na vrcholu C je *jednoduchý popisek*, a proto potřebujeme mít uloženou hodnotu tohoto popisku.

6.5.2 Ukládání XML

Aby mohl být dokument uložen do XML dokumentu, musíme vytvořit instanci třídy `XML`. Tato třída po zavolání konstruktoru

```
XML dokument = new XML();
```

vytvoří nový objekt připravený pro zápis elementů. Jelikož XML dokument popisuje příklad, jako kořenový element dokumentu je zvolen element `<priklad>`, který je rovnou v konstruktoru zapsán do struktury.

Pro zápis dalších elementů do struktury jsou v této třídě připraveny různé metody:

- `public void pridejBodA(int bodA)`
- `public void pridejBodA(int bodA, String popisek)`
- `public void pridejBodB(int bodB)`
- `public void pridejBodB(int bodB, String popisek)`
- `public void pridejBodC(int bodC)`
- `public void pridejBodC(int bodC, String popisek)`
- `public void pridejOrientaciTrojuhelniku(int orientace)`
- `public void pridejVysledekPrikladu(String vysledek)`
- `public void pridejZadaniPrikladu(String textZadani)`

Jak zde můžeme vidět, některé metody byly přetíženy, protože dopředu nevíme, zda bude typ objektu popisek nebo ne a pokud ano, tak abychom ho měli možnost korektně uložit. Všechny tyto metody jsou volány v metodě `ulozDoXML()` třídy `Editor`.

Poslední metoda související s ukládáním XML dokumentu je metoda:

```
public void ulozXML(String nazevSouboru)
```

Tato metoda provádí samostatný zápis XML dokumentu do souboru.

6.5.3 Čtení XML

Pro čtení XML souboru je definován druhý konstruktor, který je přetížený a obsahuje jeden parametr, a to cestu ke XML souboru. Získání této cesty zajišťují jiné třídy, tato třída předpokládá, že je cesta korektní.

Pro čtení XML elementů jsou definovány následující metody:

- `public int vratBodA()`
- `public int vratBodB()`
- `public int vratBodC()`
- `public int vratOrientaciTrojuhelniku()`
- `public String vratPopisek(int oblast)`
- `public String vratVysledekPrikladu()`
- `public String vratZadaniPrikladu()`

Pomocí těchto metod tedy jednoduše získáme informace o příkladu a můžeme těmito hodnotami naplnit prvky Editoru.

Kapitola 7

Možnosti rozšíření aplikací

Tato kapitola popisuje některá možná rozšíření, která by mohla v budoucnu vylepšit obě aplikace.

7.1 Lepší distribuce příkladů mezi aplikacemi

Distribuce příkladů mezi aplikacemi je zajišťována klasickým přenosem souborů na přenosných médiích, například na flash discích, nebo pomocí FTP serveru do školní sítě. Aplikace je navržena tak, aby si kantor doma připravil příklady pro výuku a zkopíroval si je na nějaké médium nebo do síťového umístění. Jakmile by se dostal ke školní sdílené síti, nahrál by soubory do tohoto umístění a žáci by si příklady otevírali z těchto sdílených disků.

Později nás však napadlo, že by bylo mnohem lepší zabudovat do obou aplikací síťového klienta. K tomuto klientovi by existovat i server běžící na školní síti, na který by se přenášely příklady, které kantor doma připravil pro výuku. Samozřejmě se kantorovi může stát, že příklady zapomene zkopírovat a nebudou tedy pro výuku dostupné. Tento způsob by zajistil, že příklady budou dostupné vždy (samozřejmě za předpokladu, že běží daný server). Server by mohl mít s klientem zajištěnou alespoň minimální úroveň zabezpečení, například na úrovni přihlašovacího jména a hesla.

7.2 Vylepšení grafů goniometrických funkcí

Další možné rozšíření, které by mohlo být implementováno, je v zobrazování grafů goniometrických funkcí. Jak již bylo zmíněno, hodnoty u grafu jsou řízeny pomocí dvou posuvníků a zajišťují proměnnou hodnotu parametrů u periody (širší a užší průběh funkce) a hodnoty (vyšší a nižší průběh funkce).

Zde by mohl být umístěn ještě třetí posuvník, který by měnil hodnotu parametru přičítaného k hodnotě x. Tím by se docílilo posunu grafu doprava nebo doleva. Tato látka sice již plně zasahuje do středoškolských osnov, ale pro zajímavost by mohla být aplikace takto rozšířena.

7.3 Počítání výsledků v aplikaci Editor

Posledním možným rozšířením, které zde zmíníme, je rozšíření aplikace Editor o formulář pro vypočítání výsledku. Jak již víme, aplikace umožňuje zadat kompletní příklad včetně všech potřebných údajů, avšak příklad musíme vypočítat ručně.

Jako rozšíření bychom zde mohli implementovat jednoduchý formulář, kde bychom zadali hodnoty, které jsou známé a zaškrtli, co chceme pomocí formuláře spočítat. Formulář by poté výsledek spočítal a automaticky ho doplnil do pole výsledek.

Kapitola 8

Závěr

Účelem této bakalářské práce bylo vytvořit výukovou aplikaci matematiky pro základní školy. Aplikace se zaměřuje na efektivní vyučování goniometrických funkcí. Před výkladem samotné látky o goniometrických funkcích je zmíněna prerekvizitní látka obsahující poznatky o Pythagorově větě. Výuková aplikace Goniometrie je interaktivní a snaží se pomoci žákům k lepšímu zapamatování těchto matematických jevů.

K lepšímu procvičení aplikace umožňuje při výuce otevírat připravené příklady přímo v aplikaci Goniometrie. Žáci mohou tyto příklady vypočítat, a když si myslí, že dosáhli správného výsledku, nechají si výsledek zobrazit od aplikace.

K tvorbě těchto příkladů slouží druhá aplikace, která je určená převážně kantorům, ale není důvod proč by jí nemohli použít i žáci. Umožňuje vytvářet typické příklady související s goniometrickými funkcemi.

Při implementaci jsme se potýkali i s některými problémy týkající se programování v jazyce Java. K řešení nám většinou vypomohla kniha [3] a webové stránky podporované firmou Sun Microsystems [2].

Literatura

[1] *Dom4j – XML parser pro jazyk Java* [online]. [cit. 2009-05-12].
URL http://www.dom4j.org/

[2] *Java Tutorial* [online]. [cit. 2009-05-12].
URL http://www.java2s.com/Tutorial/Java/CatalogJava.htm

[3] Horton, I.: *Java 5*. Birmingham: Wrox Press, 2002, ISBN 1-861005-69-5.

[4] Wikipedie: *Didaktické zásady* [online]. [cit. 2009-01-15].
URL http://cs.wikipedia.org/wiki/Didaktické_zásady